UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE CIÊNCIAS
DEPARTAMENTO DE COMPUTAÇÃO

ESTRUTURAS DE DADOS – TI0140 2022.2 T02A

TRABALHO 2: DICIONÁRIO VIA ÁRVORE AVL (2022-11-22)

PROFESSOR: PABLO MAYCKON SILVA FARIAS

1 Resumo

Você deverá implementar em C++ um dicionário baseado em árvore AVL, sem operações de remoção, conforme interface mencionada mais à frente. A sua implementação será acoplada a um programa do professor, que realizará uma bateria de testes automáticos sobre as operações de dicionário e sobre a estrutura da árvore; caso seja detectado algum problema, o professor o informará para que seja feita uma correção (desde que dentro do prazo de submissão). A implementação será entregue ao professor através de tarefa no SIGAA.

Atenção: Colabore para o bom andamento da disciplina e do aprendizado de todos, não copiando códigos de outras pessoas nem compartilhando seu próprio código, nem mesmo em versão preliminar. Em caso de dificuldade sua ou com algum colega, fale com o professor.

2 Interface do Dicionário

Você deve implementar o dicionário a partir da "interface" disponibilizada no SIGAA juntamente com este enunciado, escrevendo os métodos solicitados e adicionando o que for necessário para isso, como funções auxiliares que você considere importantes.

Um dicionário é uma associação entre *chaves* e *valores*. Uma chave identifica um elemento de forma única (um dicionário não possui chaves repetidas), e um valor é algum dado de interesse para um certo elemento. Assim, por exemplo, nós poderíamos (fora do contexto deste trabalho) construir um dicionário para armazenar dados da turma: as chaves poderiam ser os números de matrícula (que nunca são repetidos) e os valores poderiam ser dados relativos aos estudantes, como por exemplo o nome, as notas, etc.

Na prática, implementações úteis de dicionário (como `std::map`) são genéricas quanto ao tipo das chaves e ao tipo dos valores, tipicamente usando polimorfismo paramétrico. Entretanto, como nesta disciplina nós não tivemos tempo de estudar os templates de C++, então este trabalho trata de um dicionário com um tipo fixo para chaves (`double`) e outro para os valores (`float`).

Segue abaixo um exemplo de uso da implementação solicitada (para melhor compreender os detalhes, veja o arquivo da interface no SIGAA):

```
#include <iostream>

using std::cout;

#include "solucao.cpp"
```

```
int testar ()
  {
   DicioAVL D;  int i;  DicioAVL::Noh* nohs[9];

   for (i = 0; i <= 8; ++i)
     {
      DicioAVL::Noh* n = D.inserir(i, i/4);

      if (n == nullptr or n->chave != i or n->valor != i/4) return 1;

      nohs[i] = n;
     }

   for (i = 0; i <= 8; ++i)
     {
      if (D.buscar(i) != nohs[i]) return 2;
     }

   if (D.raiz->chave != 3 or D.raiz->esq->valor != 1/4 or
       D.raiz->dir->altura != 3 or D.raiz->esq->esq->esq != &D.sent)
     {
      return 3;
     }

   return 0;
  }

int main ()
  {
   if (testar() != 0) cout << "Erro nos testes básicos!\n";
   else               cout << "Testes básicos realizados com sucesso.\n";
  }
```

3 Compilação e Submissão da Solução

A sua implementação deverá ser escrita em C++17 padrão. Você deverá completar o arquivo da interface, renomeando-o para `solucao.cpp` e incluindo sua identificação no início do arquivo:

```
// NOME:      ...
// MATRÍCULA: ...
```

Eventuais comentários sobre a sua solução do trabalho podem também ser feitos no início do arquivo.

O professor incluirá o seu código num programa de testes, a ser compilado através de uma linha como:

```
g++ -Wall -Wextra -std=c++17 -pedantic -o programa  main.cpp
```

Essa compilação deve acontecer sem erros ou avisos.

Atente ao fato de que a sua solução para o trabalho consiste apenas na implementação do dicionário `DicioAVL`, e não em um programa inteiro. Naturalmente, você precisará testar o

seu dicionário antes de submetê-lo, e portanto deverá escrever uma função `main` para fazer os testes, como no código de exemplo da seção anterior; entretanto, para evitar conflito com o código do professor, você não deve enviar a sua função `main` no arquivo da solução do trabalho, ou então deve *comentar tudo o que não seja a implementação do dicionário*.

Você deve submeter sua solução para o trabalho através da tarefa cadastrada pelo professor no SIGAA. O professor procurará testar a sua solução e dar um retorno no máximo no dia útil seguinte, de forma que você possa tentar corrigir sua implementação e ressubmetê-la, desde que dentro do prazo.

Importante: Não deixe para submeter a sua solução apenas no fim do prazo, pois, se isso acontecer, é bem possível que não haja tempo para o professor lhe dar um retorno e você corrigir a sua implementação.

Em caso de dúvida, contate o professor rapidamente.

Observação: Em princípio, você não deve usar a biblioteca padrão de C++ neste trabalho, exceto pelos cabeçalhos `<new>` e `<exception>`, para o tratamento da alocação dinâmica. Caso você deseje utilizar algo a mais, contate o professor.

– Bom trabalho! –